200313904-1 (HPCO.146PA)

METHOD AND APPARATUS FOR ALLOCATING RESOURCES TO APPLICATIONS

Inventors

Xiaoyun Zhu

Cipriano A. Santos

Julie Ward Drew

Dirk Beyer

Sharad Singhal

METHOD AND APPARATUS FOR ALLOCATING RESOURCES TO APPLICATIONS

FIELD OF THE INVENTION

[0001] The present disclosure relates to allocating resources to applications.

BACKGROUND

[0002] Utility computing is viewed by many as the model of computing for the future, although the vision has been around for decades. The MULTICS project in the 1960s had the goal of developing "a new computer system specifically organized as a prototype of a computer utility," with one of its requirements being "continuous operation analogous to that of the electric power and telephone companies." In a computing utility, computing resources and capabilities are provided to people and businesses as a service.

[0003] One example of a computing utility that exists today is the Grid, which offers spare compute cycles to scientific and engineering applications. Another example is data center, where a large pool of IT resources are centrally managed to meet the needs of business critical enterprise applications such as enterprise resource planning applications, database applications, customer relationship management applications, and general e-commerce applications. There has been a wave of industrial initiatives to provide infrastructure and management support for such utilities.

[0004] A large utility computing environment can contain thousands of servers and storage devices connected through a shared high speed network fabric.

The goal is to offer "infrastructure on demand," which means compute, networking, and storage resources are provided to applications as they need them. Most of the resources will be virtualized and shared across multiple applications to achieve economies of scale and increase return on investment. The complexity of managing such an infrastructure and applications simultaneously is enormous. Automation is needed to lower operation cost and reduce human error. Well-informed capacity planning and resource provisioning are required to increase asset utilization and meet service level objectives.

SUMMARY

[0005] A method, system, and apparatus is disclosed for allocating resources to applications. Available resources of a networked computing system are determined. For each application, required resources of the application are determined. An assigned subset of the available resources for each application is determined as a function of the required resources of the application and the available resources. The function reduces communication delays between resources of the subset of the available resources in conformance with bandwidth capacity requirements of the application and in conformance with network bandwidth limitations. The applications may then be associated with the assigned subsets of resources.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] FIG. 1 is a diagram of a utility computing infrastructure according to embodiments of the invention;

[0007] FIG. 2 is a process chart illustrating inputs to a resource assignment problem according to embodiments of the invention;

[0008] FIG. 3 is a graph illustrating an application model diagram according to embodiments of the invention;

[0009] FIG. 4 is a graph illustrating component-to-file mapping according to embodiments of the invention;

[0010] FIG. 5 is a network topology map for applying resource assignment according to embodiments of the invention; and

[0011] FIG. 6 is a flowchart showing steps of resource assignment according to embodiments of the invention.

DETAILED DESCRIPTION

[0012] In the following description of various embodiments, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration various example manners by which the invention may be practiced. It is to be understood that other embodiments may be utilized, as structural and operational changes may be made without departing from the scope of the present invention.

[0013] In general, the present disclosure relates to a resource assignment problem (RAP) for a large-scale computing utility, such as an Internet data center. FIG. 1 shows a computing utility infrastructure diagram 100 according to embodiments of the present invention. The infrastructure 100 includes servers 102 and storage devices 104 connected through a shared storage area network (SAN) 106. The storage devices may be "virtualized," meaning that it may appear to the servers 102 and other network entities as if the storage devices 104 are locally connected and controlled. The storage devices 104 are actually remotely connected via the network fabric 106, and the physical components (e.g., disk arrays) of the storage devices 104 may be shared among many servers 102 at once.

[0014] The servers 102 may also be accessed via a network 108. The computing resources of the servers 102 may be virtualized over the high speed network fabric 108, such that the computing resources (e.g., processing, memory, storage) of each server 102 may be simultaneously shared by numerous applications and users. The applications may access the computing resources internally (e.g., via an intranet 110) or externally (e.g., via the Internet 112).

[0015] The goal of the utility computing infrastructure 100 is to offer "infrastructure on demand," which means that computing, networking, and storage

resources are provided to applications as they need them. Most of the resources will be virtualized and shared across multiple applications to achieve economies of scale and increase return on investment.

[0016] A large-scale utility computing infrastructure 100 may contain thousands of servers 102 and storage devices 104. The complexity of managing such an infrastructure and applications simultaneously is enormous. Automation is needed to lower operation cost and reduce human error. Well-informed capacity planning and resource provisioning are required to increase asset utilization and meet service level objectives.

[0017] When an application is deployed in a computing utility infrastructure 100, it is allocated a partition of resources in a virtual application environment to meet the specific needs of the application. As each application's real time workload varies over time, resources can be dynamically re-allocated and re-distributed among all running applications to achieve high resource utilization. In most cases, the physical identities of the allocated resources are transparent to the application due to virtualization of resources.

[0018] It is the utility provider's job to choose the right set of physical resources for each application and its components to satisfy the application's configuration and performance requirements, to avoid resource bottlenecks in the infrastructure, to achieve certain goals or enforce certain policies. This decision-making process is referred to as "resource assignment." Techniques for dealing with this process are an integral part of a resource access management framework that controls the complete lifecycle of applications' access to resources in a computing utility.

[0019] In today's data centers, resource assignment is typically done by human operators, which is slow, expensive, and error prone. As the size of future computing utilities grows to the magnitude of tens of thousands of resources, the number of possibilities to provision a given application goes far beyond the tracking ability of any human. This calls for a more systematic approach for resource assignment so that it can be automated to significantly shorten application deployment cycles and minimize operator overhead.

[0020] It will be appreciated that in the example infrastructure 100 a resource management application 114 may be used to automatically assign resources. The resource management application 114 may be used for initial resource assignments, as well as dynamically re-allocating resources in operation. The resource management application 114 may run on one or more data processing arrangements, such as a computer 116.

[0021] In general, a naïve scheme or resource assignment such as random selection or first-come-first-served may not work because there are too many consequences to any particular solution that is chosen. For instance, the compute requirements of the application may not be met by some of the servers, the latency of the application can be poor, or the cost involved may be too high, etc. In particular, since networking resources are shared among different applications and their components, it is highly likely for a network link to become a bottleneck thus degrading the performance of the applications that share this link. This assumes that network resources are not over-provisioned, and relatively high utilization on these resources is desired. Therefore, resource assignment is a highly complex problem that requires more intelligent solution techniques.

[0022] Every application to be deployed in a computing utility has high-level requirements such as number of concurrent users, number of transactions per second and infrastructure cost. Usually the mapping between these requirements and the specific identities of the resources that are used to host the application is not straightforward. This mapping may be broken down into two steps, 1) determining resource requirements, and 2) mapping those requirements to available resources.

[0023] In FIG. 2, a two-step process 200 for mapping requirements to resources according to embodiments of the present invention is shown. The first step is referred to as "application design" 204, and involves translating the application's high-level requirements 202 into an application model 206 that represents the low-level processing, communication and storage requirements on the physical resources. The application design step 204 requires domain knowledge and experience with the specific application, and typically involves benchmarking exercises.

[0024] The application model 206 is used together with an infrastructure resource model 208 as input to the next step, resource assignment 210. Resource assignment 210 involves deciding whether sufficient server and network resources exist in the infrastructure to accommodate the application's resource requirements, and if so, choosing the specific instances of resources from the infrastructure for use by the applications. If, however, resource assignment 210 decides that no sufficient resources exist, then the application is denied admission into the computing utility. The resource assignment step 210 requires knowledge of both the physical resources and application requirements contained in the application and resource models 206, 208. The resulting resource assignment decision is then fed into an application deployment engine, which configures the switches and servers and installs associated application components on the servers.

[0025] The concepts described herein are generally directed to solving the second step, resource assignment 210. The resource assignment problem is defined as follows: For a given topology of a network consisting of switches and servers with varying capabilities, and for a given application with a distributed architecture, decide which server from the physical network should be assigned to each application component, such that the traffic-weighted average inter-server distance is minimized, and the application's processing, communication and storage requirements are satisfied without exceeding network capacity limits.

[0026] An application can be characterized by a set of components that communicate with one another in a certain way. An application model diagram according to embodiments of invention is shown in FIG. 3. The application can be represented by a directed graph G(C, L) 300, where each node $c \in C$ (e.g., 302, 304) represents an application component, and each directed edge $l = (c, c') \in L$ is an ordered pair of component nodes, representing communication from component c to component c'. The matrix T is defined to characterize the traffic pattern of the application. Each element $T_{cc'}$ represents the maximum amount of traffic going from component c to component c'. $T_{cc'} = 0$ if an edge (c,c') does not exist, indicating no traffic flows from component c to component c'.

[0027] Each application component has requirements on the type of servers on which it can be hosted. Let P to be the set of server attributes (or properties) that are of interest to a particular application, such as processor type, processor speed, number of processors, memory size, disk space, and so on. Then for each attribute $p \in P$ and each application component $c \in C$, the requirement is characterized by a set $VREQ_{cp}$, which contains the permissible values of attribute p for component c. This set may be

either discrete or continuous. For example, an application component may require a server's processor architecture to be in {SPARC, PA_RISC}, and its processor speed to be in an interval [500, 1000] (in MHz).

[0028] Referring now to FIG. 4, a model for storage requirements is illustrated according to embodiments of the invention. The storage access pattern of applications can be represented by a bipartite graph 400. It may be assumed that data for an application can be divided into a set of "files" (e.g., 402, 404). Here a file may represent any logically contiguous chunk of data that may be accessed by application components (e.g., 406, 408). The example illustrates that the mapping between an application component and a file is not one-to-one. More specifically, each component may access multiple files, and each file may be accessed by more than one component.

[0029] The above application model can be used for simultaneous assignment of resources to multiple applications. A single large graph can be constructed with all the components from all the applications, where each application is represented by a sub-graph.

[0030] With this in mind, the application model contains the sets and parameters shown below in Table 1.

Sets and Indices:

$c \in C$	Set of application components.
$f \in F$	Set of files to be placed on storage devices.
$l \in L$	Set of directed links in the application architecture graph.
$c' \in N_c$	Set of components that communicate with component c, i.e., $N_c = \{c' \in C : (c,c') \in L\}$.
$p \in P$	: Set of server attributes required by the application.

Parameters:

T	$\|C\| \times \|C\|$-dim matrix. $T_{cc'}$ is the amount of traffic from component c to component c'.
TCF	$\|C\| \times \|F\|$-dim matrix. TCF_{cf} is the amount of write traffic from component c to file f.
TFC	$\|F\| \times \|C\|$-dim matrix. TFC_{fc} is the amount of read traffic from file f to component c.
TO	$\|C\|$-dim vector. $TO_c = \sum_{c' \in N_c} T_{cc'}$ is the total amount of LAN traffic originating from component c.
TI	$\|C\|$-dim vector. $TI_c = \sum_{c' \in N_c} T_{c'c}$ is the total amount of LAN traffic received by component c.
$VREQ_{cp}$:	The set of the permissible values of attribute p for component c.

Table 1

[0031] The following paragraphs describe the mathematical models for the processing, networking and storage resources in a computing utility. The collection of resources as a whole is referred to as the "utility fabric", which includes servers that can be assigned to applications, the local area networking (LAN) fabric (e.g., Ethernet) that connects the servers to each other, and the storage area network (SAN) fabric that connects the servers to the centralized storage devices.

[0032] Let S be the set of servers in the physical network. The notion of a "server" here is not restricted to a compute server. It can be a firewall, a load balancer, a network attached storage (NAS) device, a VPN gateway, or any other device an application may need as a component. An attribute "server type" is used to distinguish between different kinds of servers. Due to the inherent heterogeneity of resources in a large computing utility, even the same type of servers may have different processor architecture and processing

power. Therefore, more attributes are used to describe a server. The value for each attribute may be fixed, or configurable. For example, a server may have a server may have an "IA32" architecture, a CPU speed of 550 MHZ, but its memory size is changeable between 4 and 8 MB. For each server $s \in S$, the set V_{sp} is used to represent its possible values for attribute $p \in P$.

[0033] Before describing the mathematical models for the networking fabric, a common set of networking assumptions may be made to simplify the models. All the network links are assumed to be duplex links and traffic can flow in either direction. In addition, link capacities for the two directions can be different. For any physical link in any direction, its "link capacity" is indeed the minimum of the bandwidth capacities of the link, the source port and the destination port.

[0034] Multiple physical links between two devices that are all active and load balanced are combined into one logical link with aggregated capacity. For example, four 1 Gbit/sec physical links can be combined to form one 4 Gbit/sec link in the logical topology. This simplification is valid when the combined links have equal bandwidth and share approximately equal load, which is typically true. This is also the case if trunking technology is applied on the links.

[0035] If two switches appear in a redundant pair to avoid single point of failure, then redundant paths exist between at least one pair of devices in the physical topology. This can be simplified in different ways depending on the network protocol the switches implement. For example, in the LAN fabric, the spanning tree protocol may be enforced, resulting in all the redundant paths between two network devices being blocked except one. If two switches in a redundant pair are both active and being load balanced, then the switches or servers that are connected to these two

switches can be partitioned into two sets, one under each switch. And the cross links will be blocked.

[0036] Similarly, the SAN fabric may implement the Fabric Shortest Path First (FSPF) protocol, which assures uniform traffic load sharing over equivalent paths. Moreover, the two links in the same segment of the two paths usually have the same bandwidth. As a consequence, a pair of redundant switches can be merged into one switch. Corresponding links will also be merged to form a bigger link with aggregated bandwidth.

[0037] These simplifying assumptions may be applied to both the LAN and the SAN fabrics as they are represented using mathematical models. It may be assumed that the logical topology of the LAN fabric in the computing utility is a tree. This is a reasonable assumption given that a layer-two switched network often implements the spanning tree protocol, guaranteeing that there is one and only one active path between two network devices. The tree network topology significantly simplifies the formulation of the problem later on.

[0038] In reference now to FIG. 5, an example of the LAN fabric topology 500 is shown according to various embodiments of the invention. At the top is a switching/routing device 502 that connects the utility fabric to the Internet or other utility fabrics. This device 502 may be referred to as a root switch. Below the root switch 502 is a set of edge switches 504, and below the edge switches 504 is a set of rack switches 506. Servers 508 are directly connected to either an edge switch 504 or a rack switch 506. As the figure shows, an edge switch 504 can be connected to a set of rack switches 506, a set of servers 508, or a combination of both.

[0039] The three-layer network shown in FIG. 5 is chosen for demonstration purposes. It will be appreciated that the models described herein may be adapted for

any LAN fabric topology that can be represented as a tree. Therefore the methodology described herein may be applied to a tree network with fewer layers or more layers.

[0040] The mathematical model for the LAN contains the following sets and parameters shown below in Table 2.

Sets and Indices

$s \in S$	Set of servers.
$r \in R$	Set of rack switches in the LAN.
$e \in E$	Set of edge switches in the LAN.
$R_e \subset R$	Set of rack switches connected to edge switch e in the LAN.
$SR_r \subset S$	Set of servers connected to LAN rack switch r.
$SE_e \subset S$	Set of servers connected (directly or indirectly) under LAN edge switch e.
$p \in P$:	Set of server attributes required by the application.

Parameters:

BSI_s	The incoming bandwidth of server s.
BSO_s	The outgoing bandwidth of server s.
BRI_r	The incoming bandwidth of rack switch r.
BRO_r	The outgoing bandwidth of rack switch r.
BEI_e	The incoming bandwidth of edge switch e.
BEO_e	The outgoing bandwidth of edge switch e. V_{sp} : Set of possible values for attribute p of server s.

Table 2

[0041] For easy indexing, each logical link in the network is associated with a device with which it may be uniquely identified. For example, the link that connects server s to a rack or edge switch is associated with that server and its

downstream/upstream bandwidth is referred to as the incoming/outgoing bandwidth of server *s*. The same rule applies to the links at the upper layers.

[0042] Various SAN topologies have been used in practice. The popular ones include ring, cascade, mesh, and core/edge topologies. Among these, the core/edge topology provides better resiliency, scalability, flexibility and throughput, and is adopted by many vendors and SAN designers. Therefore, it will be assumed that the SAN fabric in a computing utility has a core/edge topology. The lower portion of FIG. 5 exemplifies a SAN with this topology.

[0043] The core/edge topology contains two layers of switches. The core layer consists of at least one pair of redundant core switches 512 that are typically the most powerful. All the other switches connected to the core switches 512 are referred to as edge switches 510. The centralized storage devices 514, such as disk arrays, are attached directly to the core switches 512, and the servers 508 are attached directly to the edge switches 510. The above topology ensures that every storage device 514 is accessible by any server 508 in the SAN. Note that this logical topology is a simplification from the physical topology with redundancies in network devices and links.

[0044] The mathematical model for the SAN contains sets and parameters shown below in Table 3.

Sets and indices:

$s \in S$	Set of servers.
$d \in D$	Set of storage devices.
$k \in K$	Set of FC core switches in the SAN.
$g \in G$	Set of FC edge switches in the SAN.
$SED_g \subset S$	Set of servers connected to FC edge switch g.
$SCO_k \subset S$	Set of servers (indirectly) connected to FC core switch k.

Parameters:

BDC	$\|D\| \times \|K\|$-dim matrix. BDC_{dk} is the bandwidth of the FC link going from storage device *d* to core switch *k*.
BCD	$\|K\| \times \|D\|$-dim matrix. BCD_{kd} is the bandwidth of the FC link going from core switch *k* to storage device *d*.
BCE	$\|G\|$-dim vector. BCE_g is the bandwidth of the FC link going from a core switch to edge switch *g*.
BEC	$\|G\|$-dim vector. BEC_g is the bandwidth of the FC link going from edge switch *g* to a core switch.
BES	$\|S\|$-dim vector. BES_s is the bandwidth of the FC link going from an edge switch to server *s*.
BSE	$\|S\|$-dim vector. BSE_s is the bandwidth of the FC link going from server *s* to an edge switch.

Table 3

[0045] The resource assignment problem concerns selecting the right server in the utility fabric for each application component, represented by the following matrix of binary variables: For all $c \in C$ and $s \in S$,

$$x_{cs} = \begin{cases} 1 & \text{server s assigned to component c;} \\ 0 & \text{otherwise.} \end{cases}$$

[0046] In addition, the following two matrices of binary variables are defined. For all $c \in C$, $r \in R$, and $e \in E$,

$$zr_{cr} = \begin{cases} 1 & \text{rack switch r assigned to component c;} \\ 0 & \text{otherwise.} \end{cases}$$

$$ze_{ce} = \begin{cases} 1 & \text{edge switch e assigned to component c;} \\ 0 & \text{otherwise.} \end{cases}$$

[0047] It may be assumed a switch is assigned to a component if at least one server connected (directly or indirectly) under the switch is assigned to that component. Note that these two variables are redundant to the variables x_{cs}. They are introduced to help express the Ethernet bandwidth constraints in a more succinct way, and to make solving of the problem more efficient.

[0048] Resources in a computing utility can be assigned to application components based on many criteria, such as application performance, resource utilization, operator policies, or economic concerns. These can be associated with different objective functions of the optimization problem. As formulated herein, the objective function used in the node placement optimization problem is chosen, which minimizes the traffic-weighted average inter-server distance where distance is measured in terms of network hop count. Let $DIST_{ss'}$ be the distance between two servers s and s', and $TSS_{ss'}$ be the amount of LAN traffic from server s to server s' as a result of server assignment. Then the objective function is:

$$\text{Min} \quad J1 = \sum_{s,s' \in S} DIST_{ss'} * TSS_{ss'} \ .$$

[0049] As may be apparent, $TSS_{ss'} = \sum_{c \in C} \sum_{c' \in N_c} x_{cs} T_{cc'} x_{c's'}$. The value of $DIST_{ss'}$ depends on the relative location of server s and s'. For example, $DIST_{ss'} = 2$ if both servers are directly connected to the same switch, which is a preferred situation if these two servers communicate heavily.

[0050] By dividing the set of all server pairs into a number of subsets, each with a different $DIST_{ss'}$ value, then calculating the summation on each subset and adding them up, this results in:

$$J1 = 2\sum_{c\in C}(TO_c + TI_c) + \sum_{r\in R}\sum_{c\in C} zr_{cr}(TO_c + TI_c)$$
$$-2\sum_{r\in R}\sum_{c\in C}\sum_{c'\in N_c} zr_{cr}T_{cc'}zr_{c'r} - \sum_{e\in E}\sum_{c\in C}\sum_{c'\in Nc} 2ze_{ce}T_{cc'}ze_{c'e}$$

[0051] The first term is the total amount of traffic originated from and received by all the components, which is a constant. Therefore, an equivalent objective function follows:

$$\text{Min} \quad J2 = \sum_{r\in R}\sum_{c\in C} zr_{cr}(TO_c + TI_c)$$
$$-2\sum_{r\in R}\sum_{c\in C}\sum_{c'\in N_c} zr_{cr}T_{cc'}zr_{c'r} - \sum_{e\in E}\sum_{c\in C}\sum_{c'\in Nc} 2ze_{ce}T_{cc'}ze_{c'e}$$

[0052] This is a quadratic function of the binary variables zr_{cr} and ze_{ce}. The first term represents the total amount of traffic originated and received under all the rack switches. A similar term for all the edge switches, $\sum_{e\in E}\sum_{c\in C} ze_{ce}(TO_c + TI_c)$, would have been present, but was removed as part of the constant term. The second and third terms together capture the total amount of intra-switch traffic at all the switches. Here "intra-switch traffic" is defined as the traffic flows whose source and destination nodes are servers under the same switch. The intuition is, as components that communicate heavily are placed close to each other in the network, the amount of intra-switch traffic is increased, which in turn results in smaller value for the objective function. In general, this leads to lower communication delay between application components inside the LAN fabric.

[0053] SAN latency is not included in the objective function for the following two reasons. First, the SAN topology in this problem has the property that the number of hops for each data flow is fixed at three because any server and storage device pair is connected through two FC switches. This means, any server assignment solution results in the same SAN latency measure. Second, storage systems latency is

dominated by I/O access at the storage device, which is typically several orders of magnitude larger than the SAN latency. Therefore, even if the number of hops could be reduced between a server and a storage device, it is inconsequential with respect to storage access latency. On the other hand, link capacity in the SAN is usually a concern in storage systems performance. Given the high cost of SAN switches, grossly over-provisioning may not be preferred, while at the same time it is not desirable to allow the SAN fabric to be easily saturated. With this observation, the SAN link capacity in RAP is handled without adding any new objective function. The rest of this section describes constraints in the problem that limit the search space for optimal server assignment solutions.

[0054] Before describing constraints in the RAP, a server feasibility matrix FS is defined, where:

$$FS_{cs} = \begin{cases} 1 & \text{switch s meets the processing, networking,} \\ & \text{and I/O requirements of component c;} \\ 0 & \text{otherwise.} \end{cases}$$

[0055] More specifically, $FS_{cs} = 1$ if and only if

$$V_{sp} \cap VREQ_{cp} \neq \phi, \quad \forall p \in P \qquad \{a\}$$

$$\sum_{c' \in N_c} T_{c'c} \leq BSI_s \text{ and } \sum_{c' \in N_c} T_{cc'} \leq BSO_s \qquad \{b\}$$

$$\sum_{f \in F} TCF_{cf} \leq BSE_s \text{ and } \sum_{f \in F} TFC_{cf} \leq BES_s \qquad \{c\}$$

[0056] Condition {a} ensures that server *s* matches the server attribute requirement by component *c*. Condition {b} ensures that the aggregate LAN traffic at each component *c* does not exceed the link bandwidth of server *s* in either direction. And condition {c} guarantees that the total amount of SAN traffic at each component *c* does not exceed the I/O bandwidth of server *s* in either direction.

[0057] The server feasibility matrix can be pre-computed before the optimization problem is solved. When the matrix FS is sparse, the search space for the optimization problem can be significantly reduced.

[0058] Similarly, feasibility matrices FR and FE can be defined for rack and edge switches, respectively, where $FR_{cr} = 1$ if there is at least one feasible server under rack switch r for component c, $FE_{ce} = 1$ if there is at least one feasible server under edge switch e for component c. These two matrices can also be pre-computed.

[0059] The constraints on the decision variables are as follows.

[0060] Normality constraints: One and only one server is assigned to each application component:

$$\sum_{s \in S} x_{cs} = 1, \quad \forall c \in C. \qquad \{1\}$$

[0061] Each server can be assigned to at most one component:

$$\sum_{c \in C} x_{cs} \leq 1, \quad \forall s \in S. \qquad \{2\}$$

[0062] Variable relationship constraints: A rack switch is assigned to a component if and only if a server under this rack switch is assigned to this component:

$$\sum_{s \in SR_r} x_{cs} = zr_{cr}, \quad \forall c \in C, \ r \in R. \qquad \{3\}$$

[0063] An edge switch is assigned to a component if and only if a server under this edge switch is assigned to this component:

$$\sum_{s \in SE_e} x_{cs} = ze_{ce}, \quad \forall c \in C, \ e \in E. \qquad \{4\}$$

[0064] LAN fabric constraints: The LAN traffic going out of each rack switch to an edge switch does not exceed the link capacity:

$$\sum_{c \in C} TO_c zr_{cr} - \sum_{c \in C} \sum_{c' \in N_c} zr_{cr} T_{cc'} zr_{c'r} \leq BRO_r, \quad \forall r \in R. \quad \{5\}$$

[0065] Remember that TO_c is the total amount of LAN traffic originating from component *c*. On the left hand side, the first item represents the total amount of traffic originating under rack switch *r*, and the second item represents the amount of intra-switch traffic at this switch. Hence, the left hand side represents the amount of traffic passing through switch *r*, which should be bounded by the outgoing link bandwidth at the switch.

[0066] The derivation of the following three constraints is similar, therefore will be omitted. The LAN traffic coming into each rack switch from an edge switch does not exceed the link capacity:

$$\sum_{c \in C} TI_c zr_{cr} - \sum_{c \in C} \sum_{c' \in N_c} zr_{cr} T_{cc'} zr_{c'r} \leq BRI_r, \quad \forall r \in R \qquad \{6\}$$

[0067] Remember that TI_c is the total amount of LAN traffic received by component *c*.

[0068] The LAN traffic going out of each edge switch to the root switch does not exceed the link capacity:

$$\sum_{c \in C} TO_c ze_{c,e} - \sum_{c \in C} \sum_{c' \in Nc} ze_{ce} T_{cc'} ze_{c'e} \leq BEO_e, \quad \forall e \in E. \quad \{7\}$$

[0069] The LAN traffic coming into each edge switch from the root switch does not exceed the link capacity:

$$\sum_{c \in C} TI_c ze_{ce} - \sum_{c \in C} \sum_{c' \in Nc} ze_{ce} T_{cc'} ze_{c'e} \leq BEI_e, \quad \forall e \in E. \qquad \{8\}$$

[0070] SAN fabric constraints: The SAN traffic going out of each FC edge switch to a core switch does not exceed the link capacity:

$$\sum_{s \in SED_g} \sum_{f \in F} \sum_{c \in C} TCF_{cf} x_{cs} \leq BEC_g, \quad \forall g \in G. \qquad \{9\}$$

[0071] The SAN traffic coming into each FC edge switch from a core switch does not exceed the link capacity:

$$\sum_{s \in SED_g} \sum_{f \in F} \sum_{c \in C} TFC_{fc} x_{cs} \le BCE_g, \quad \forall g \in G. \qquad \{10\}$$

[0072] The SAN traffic from an FC core switch to a storage device does not exceed the link capacity:

$$\sum_{s \in SCO_k} \sum_{f \in F} \sum_{c \in C} TCF_{cf} x_{cs} Y_{fd} \le BCD_{kd}, \quad \forall k \in K, \; d \in D. \qquad \{11\}$$

[0073] Here Y_{fd} is a binary parameter, where $Y_{fd} = 1$ if and only if file f is placed on storage device d. The file placement problem can be separated from the server assignment problem. The former has Y_{fd} as its decision variable. The solution is fed into the RAP problem as an input.

[0074] The SAN traffic from a storage device to an FC core switch does not exceed the link capacity.

$$\sum_{s \in SCO_k} \sum_{f \in F} \sum_{c \in C} TFC_{fc} x_{cs} Y_{fd} \le BDC_{dk}, \quad \forall k \in K, \; d \in D \; \{12\}$$

[0075] Feasibility constraints: All the variables are binary, and all the assigned servers, rack switches, and edge switches are feasible.

$$x_{cs} \in \{0, FS_{cs}\}, zr_{cr} \in \{0, FR_{cr}\}, ze_{ce} \in \{0, FE_{ce}\} \; \{13\}$$

[0076] In summary, the complete formulation of the optimization problem for RAP is

$$\text{Min} \quad J2 = \sum_{r \in R} \sum_{c \in C} zr_{cr}(TO_c + TI_c)$$
$$- 2 \sum_{r \in R} \sum_{c \in C} \sum_{c' \in N_c} zr_{cr} T_{cc'} zr_{c'r} - \sum_{e \in E} \sum_{c \in C} \sum_{c' \in Nc} 2 ze_{ce} T_{cc'} ze_{c'e}$$

[0077] subject to {1} – {13} above. This is a nonlinear combinatorial optimization problem, which has been proven as NP-hard. This problem is referred to

as the original formulation of RAP and labeled as RAP0. The problem formulation described above can be applied to a number of different use cases, some of which are shown in Table 4.

Use Case	Description
Green-field assignment	This occurs when the first application is initially deployed in an empty utility.
Subsequent assignment	This occurs when there are existing applications running in the utility, and resources are assigned to the next application. In this case, the same application and resource models can be used, except that parameters in the resource model should reflect the remaining resource capacity.
Multiple applications assignment	This occurs when resources need to be assigned to more than one application at the same time. A larger application model with components from multiple applications can be used for this purpose.
Dynamic assignment	This occurs when an existing application requests for more resources as its real time workload intensity changes. In this case, a new application model will be submitted containing the additional requirement. Depending on the application's ability to accommodate server migration, the problem can be resolved with or without fixing the existing server assignment.
Automatic fail over	This occurs when a server without high-availability configuration fails and needs replacement. The best server to use from the pool of available servers can be found using a similar RAP formulation

Table 4

[0078] The first three use cases happen at application deployment time, while the last two use cases are useful at run time. Therefore, the former is at a time scale of days or longer, while the latter may be at a shorter time scale of minutes or hours.

[0079] The number of binary variables in RAP0 is $|C|\times(|S|+|R|+|E|)$, which is dominated by $|C|\times|S|$, the number of application components times the number of servers in the utility. It is conceivable that the problem becomes computationally more challenging as the infrastructure size or application size grows. Any heuristic search algorithms are not guaranteed to find a feasible and optimal solution. The next section presents two linearized formulations as mixed integer programming problems, which can be solved directly using a commercial solver, such as CPLEX.

[0080] As previously described, the original formulation RAP0 is nonlinear because the objective function and the LAN fabric constraints {5}-{8} are quadratic in binary variables zr_{cr} and ze_{ce}. This type of nonlinearity can be removed using a standard substitution technique with the observation that the product of binary variables is also binary. First, the following set of binary variables are defined, $yr_{cc'r} = zr_{cr}zr_{c'r}$ and $ye_{cc'e} = ze_{ce}ze_{c'e}$, for all $c,c' \in C$, $r \in R$, $e \in E$.

[0081] With these new variables, the objective function can be rewritten as

$$\text{Min} \quad J2 = \sum_{r\in R}\sum_{c\in C} zr_{cr}(TO_c + TI_c)$$
$$-2\sum_{r\in R}\sum_{c\in C}\sum_{c'\in Nc} T_{cc'}yr_{cc'r} - 2\sum_{e\in E}\sum_{c\in C}\sum_{c'\in Nc} T_{cc'}ye_{cc'e}$$

[0082] This is a linear combination of all the zr_{cr}, $yr_{cc'r}$ and $ye_{cc'e}$ variables. Similarly, constraints {5} through {8} in RAP0 can be rewritten as linear constraints as follows:

$$\sum_{c\in C} TO_c zr_{cr} - \sum_{c\in C}\sum_{c'\in Nc} T_{cc'} yr_{cc'r} \le BRO_r, \quad \forall r \in R \qquad \{5I\}$$

$$\sum_{c\in C} TI_c zr_{cr} - \sum_{c\in C}\sum_{c'\in Nc} T_{cc'} yr_{cc'r} \le BRI_r, \quad \forall r \in R \qquad \{6I\}$$

$$\sum_{c\in C} TO_c ze_{ce} - \sum_{c\in C}\sum_{c'\in Nc} T_{cc'} ye_{cc'e} \le BEO_e, \quad \forall e \in E \qquad \{7I\}$$

$$\sum_{c\in C} TI_c ze_{ce} - \sum_{c\in C}\sum_{c'\in Nc} T_{cc'} ye_{cc'e} \le BEI_e, \quad \forall e \in E. \qquad \{8I\}$$

[0083] Additional constraints are used to ensure that the $yr_{cc'r}$ variables behave as the product of binary variables. First, to ensure that $zr_{cr} = 0$ or $zr_{c'r} = 0 \Rightarrow yr_{cc'r} = 0$, the following is used:

$$zr_{cr} \ge yr_{cc'r}, \quad zr_{c'r} \ge yr_{cc'r} \quad \forall c,c' \in C, r \in R. \qquad \{13I\}$$

[0084] Second, to ensure $zr_{cr} = 1$ and $zr_{c'r} = 1 \Rightarrow yr_{cc'r} = 1$, the following constraint is used:

$$zr_{cr} + zr_{c'r} - yr_{cc'r} \le 1 \quad \forall c,c' \in C,\ r \in R.$$

[0085] However, since the objective function is to maximize a summation of the $yr_{cc'r}$ variables with non-negative coefficients, the second set of constraints are implied by the first set of constraints at optimality, and therefore are not required. Similarly, the following set of constraints should be imposed on the new $ye_{cc'e}$ variables:

$$ze_{ce} \ge ye_{cc'e}, \quad ze_{c'e} \ge ye_{cc'e} \quad \forall c,c' \in C, e \in E.$$

[0086] Note that the new $yr_{cc'r}$ and $ye_{cc'e}$ variables only need to be continuous in the interval [0,1] instead of being binary. For example, based on the above discussion, constraint {13I} and the maximization nature of the objective function together ensure that $yr_{cc'r}$ behaves exactly as the product of zr_{cr} and $zr_{c'r}$. Since

$zr_{c'r}$ and zr_{cr} are both binary, $yr_{cc'r}$ never really takes a fractional value between 0 and 1.

[0087] The above substitution of variables results in a linear optimization problem with some integer variables and some continuous variables, thus a mixed integer programming problem. It is referred to as RAP-LINI, to be distinguished from the original nonlinear formulation RAP0. The main issue with this formulation is that the number of variables may be significantly higher than that of RAP0 with the introduction of $|C|\times|C|\times(|R|+|E|)$ continuous variables. There are a number of ways to improve the efficiency in solving the problem.

[0088] First, the number of $yr_{cc'r}$ and $ye_{cc'e}$ variables can be reduced in the following way: $yr_{cc'r}$ is defined if and only if $FR_{cr}=1, FR_{c'r}=1$, and $T_{cc'}>0$; and $ye_{cc'e}$ is defined if and only if $FE_{ce}=1, FE_{c'e}=1$, and $T_{cc'}>0$. In all the other cases, the $yr_{cc'r}$ and $ye_{cc'e}$ variables are not needed in the formulation. This implies that, in the worst case where all the rack and edge switches are feasible for all the components, the number of extra variables in RAP-LINI is $|L|\times(|R|+|E|)$, i.e., the number of communication links in the application graph times the total number of LAN switches.

[0089] A second way of improving efficiency is to realize that, since the number of zr_{cr} and ze_{ce} variables ($|C|\times(|R|+|E|)$) is usually significantly less than the number of x_{cs} variables $|C|\times|S|$, the efficiency of the branch and bound algorithm in the MIP solver can be increased by assigning higher priority to branching on variables ze_{ce} and zr_{cr}.

[0090] The RAP-LINI uses a linearization technique that is straightforward and that results in a MIP formulation with $|L|\times(|R|+|E|)$ additional continuous

variables than RAP0. This subsection describes a relatively more sophisticated linearization scheme, which leads to another MIP formulation with possibly fewer extra variables.

[0091] When looking at the LAN traffic flowing through each rack switch, it will be appreciated that, for all $c \in C$ and $r \in R$, $zr_{cr}TO_c$, is the amount of traffic originating from component c under switch r, and $\sum_{c' \in N_c} zr_{c'r}T_{cc'}$ is the amount of traffic originating from component c and received under switch r. Now a define a new variable, $tro_{cr} = zr_{cr}TO_c - zr_{cr}\sum_{c' \in N_c} zr_{c'r}T_{cc'}$, which captures the amount of traffic that originated from component c under switch r and leaves switch r.

[0092] By definition of zr_{cr},

$$tro_{cr} = \begin{cases} zr_{cr}TO_c - \sum_{c' \in N_c} zr_{c'r}T_{cc'}, & \text{if } zr_{cr} = 1; \\ 0, & \text{if } zr_{cr} = 0. \end{cases}$$

[0093] Therefore, tro_{cr} can be equivalently defined as,

$$tro_{cr} = \max\left\{zr_{cr}TO_c - \sum_{c' \in N_c} zr_{c'r}T_{cc'}, 0\right\}$$

[0094] Since tro_{cr} represents the amount of outgoing traffic from component c that passes through rack switch r, and the objective function tends to reduce the amount of traffic that passes through switches, the above definition can be enforced using the following two linear constraints:

$$tro_{cr} \geq zr_{cr}TO_c - \sum_{c' \in N_c} zr_{c'r}T_{cc'} \text{ and } tro_{cr} \geq 0 \qquad \{13II\}$$

[0095] That is, these constraints will be binding at optimality.

[0096] Using the new variables tro_{cr}, the rack switch outgoing bandwidth constraint {5} in RAP0 can be rewritten as

$$\sum_{c \in C} tro_{cr} \leq BRO_r, \quad \forall r \in R \qquad \{5II\}$$

[0097] Similarly, the amount of LAN traffic originating from component c that leaves edge switch e can be represented using the following new variable: $teo_{ce} = ze_{ce} TO_c - ze_{ce} \sum_{c' \in N_c} ze_{c'e} T_{cc'}$. This would be enforced by the following constraints:

$$teo_{ce} \geq ze_{ce} TO_c - \sum_{c' \in N_c} ze_{c'e} T_{cc'} \text{ and } teo_{ce} \geq 0 \quad \{15II\}$$

[0098] Then constraint {7} of RAP0 can be rewritten as

$$\sum_{c \in C} teo_{ce} \leq BEO_e, \quad \forall e \in E \qquad \{7II\}$$

[0100] Analogous variables tri_{cr} (tei_{ce}) representing the amount of incoming traffic to component c under rack switch r (edge switch e) from components outside the switch can be defined, with the following additional constraints:

$$tri_{cr} \geq zr_{cr} TI_c - \sum_{c' \in N_c} zr_{c'r} T_{c'c} \text{ and } tri_{cr} \geq 0 \qquad \{14II\}$$

$$tei_{ce} \geq ze_{ce} TI_c - \sum_{c' \in N_c} ze_{c'e} T_{c'c} \text{ and } tei_{c,e} \geq 0 \quad \{16II\}$$

[0101] Then constraints {16} and {18} of RAP0 can be rewritten as

$$\sum_{c \in C} tri_{cr} \leq BRI_r, \quad \forall r \in R \qquad \{6II\}$$

$$\sum_{c \in C} tei_{ce} \leq BRI_e, \quad \forall e \in E \qquad \{8II\}$$

[0102] By comparing the definition of the new variables with the objective function J2 in RAP0, it can be seen that,

$$J2 = \sum_{r \in R} \sum_{c \in C} (tro_{cr} + tri_{cr}) + \sum_{e \in E} \sum_{c \in C} (teo_{ce} + tei_{ce}) - \sum_{e \in E} \sum_{c \in C} ze_{ce} (TO_c + TI_c)$$

[0103] Since $\sum_{e\in E}\sum_{c\in C} ze_{ce}(TO_c + TI_c) = \sum_{c\in C}(TO_c + TI_c)$ is a constant, an equivalent objective function is the following.

$$\text{Min} \quad J3 = \sum_{r\in R}\sum_{c\in C}(tro_{cr} + tri_{cr}) + \sum_{e\in E}\sum_{c\in C}(teo_{ce} + tei_{ce})$$

[0104] The interpretation of the objective function follows. To reduce the traffic-weighted average inter-server distance, it is equivalent to minimize the total amount of traffic flowing on all the Ethernet links. Because the total amount of traffic originating from and received by all the application components is a constant, the total amount of traffic flowing on all the server-to-switch links is a constant. Therefore, an equivalent objective function is to minimize the total amount of inter-switch traffic, which is exactly what J3 is. The term "inter-switch traffic" refers to the traffic flowing on a link that connects two switches. These links are typically more expensive. And they are more likely to get saturated because they are often shared by multiple components, or even multiple applications. By minimizing the utilization of these shared links by a single application, the likelihood of creating bottlenecks in the LAN fabric is decreased.

[0105] This MIP formulation of the resource assignment problem is referred to as RAP-LINII. In this case, a total number of $2|C|\times(|R|+|E|)$ new continuous variables are introduced. This approach involves fewer extra variables than the RAP-LINI approach if $2|C| < |L|$, i.e., if each application component has, on average, more than 2 incident links.

[0106] In case studies performed on the two mixed-integer processing formulations (RAP-LINI, RAP-LINII), the RAP-LINII formulation was found to be more efficient.

[0107] In reference now to FIG. 6, a flowchart 600 shows steps in performing the resource assignment according to embodiments of the invention. The "application design" step (602) may first be performed, which involves determining for each application a set of processing and storage resources required by the application. The system parameters are also determined (604), including available process resources, storage resources, and capacities of network data links. These resources may be considered constant or variable depending on the application (e.g., application deployment time versus automatic fail-over).

[0108] Once the application and network resources have been defined, the resource assignment problem can be solved (606). This typically involves determining an assigned subset of the available resources as a function of the application resource requirements and the available resources. The solution may involve minimizing communication delays between resources, satisfying server attribute and bandwidth capacity requirements of the application, and satisfying network bandwidth limits. The solution (606) may utilize any of the described formulations for linearizing the Ethernet fabric constraints (e.g., RAP-LINI, RAP-LINII). The formulation may be chosen based on computing efficiency. Finally, the solution obtained is used to associate (608) the applications with the assigned subset of resources.

[0109] From the description provided herein, those skilled in the art are readily able to combine hardware and/or software created as described with appropriate general purpose or system and/or computer subcomponents embodiments of the invention, and to create a system and/or computer subcomponents for carrying out the method embodiments of the invention. Embodiments of the present invention may be implemented in any combination of hardware and software.

[0110] The foregoing description of the example embodiments of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention not be limited with this detailed description, but rather the scope of the invention is defined by the claims appended hereto.